

FIG. 1

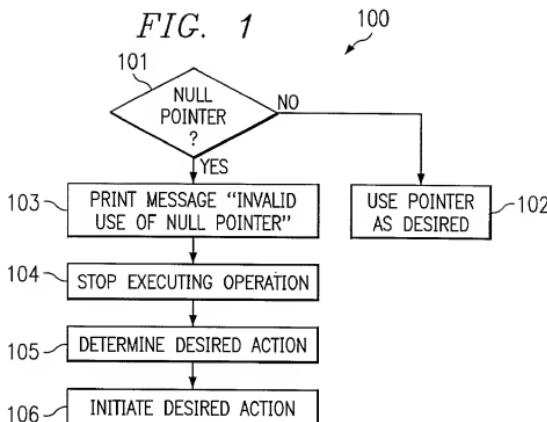


FIG. 2

```

201 #include <iostream.h>
203 template <class C>
204 class SafePtr {
205 public:
206     SafePtr (C *p) : ptr(p) {} 207     208
209     operator C *() const { return ptr; } 210
211     C *operator-> () const {
212         if (!ptr) {
213             cerr << "\nAttempted indirection of "
214             << "NULL pointer.\n";
215             exit(1); // or throw an exception
216         }
217     }
218     C operator* () const { return *operator->(); }
219     C *const ptr;
220 };
  
```

The code defines a template class SafePtr. It has a public constructor SafePtr(C *p) that initializes a member variable ptr to p. It has a public operator C *() const that returns the value of ptr. It has a public operator C *const operator-> () const that returns the value of operator->(). If ptr is null, it prints an error message and exits. It also has a private member C *const ptr.

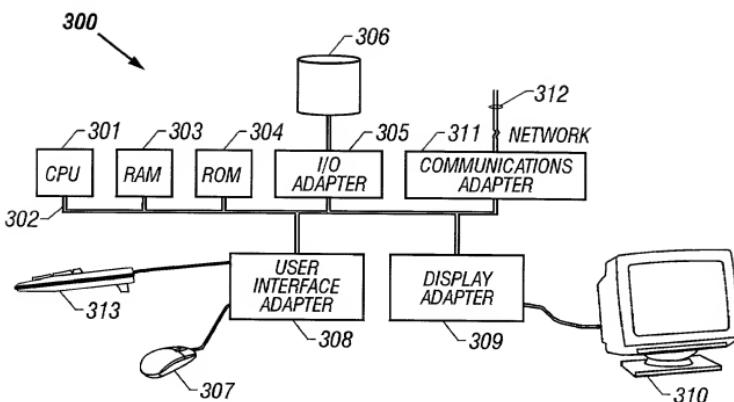


FIG. 3